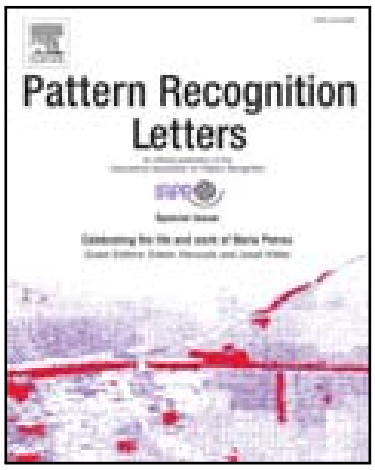

Compression and Registration of 3D Point Clouds Using GMMs

Javier Navarrete, Diego Viejo, Miguel Cazorla

PII: S0167-8655(18)30098-9
DOI: 10.1016/j.patrec.2018.03.017
Reference: PATREC 7112

To appear in: *Pattern Recognition Letters*

Received date: 11 July 2017
Revised date: 1 March 2018
Accepted date: 18 March 2018

Please cite this article as: Javier Navarrete, Diego Viejo, Miguel Cazorla, Compression and Registration of 3D Point Clouds Using GMMs, *Pattern Recognition Letters* (2018), doi: 10.1016/j.patrec.2018.03.017

Highlights

- Our main goal is to perform a combined method which compresses 3D data and could also be used for 3D registration.
- The 3D compression method uses planar patches and GMMs and outperforms state-of-theart methods
- The 3D registration method is able to use the compression data to make the registration.
- The combined compression and registration is interesting because compressed data has not to be decompressed.

Pattern Recognition Letters
journal homepage: www.elsevier.com

Compression and Registration of 3D Point Clouds Using GMMs

Javier Navarrete[a], Diego Viejo[a], Miguel Cazorla[a,**]

[a]*Instituto de Investigación en Informática*
University of Alicante
PO. Box 99, E-03080 Alicante, Spain

ABSTRACT

3D data sensors provide an enormous amount of information. It is necessary to develop efficient methods to manage this information under certain time, bandwidth or storage space requirements. In this work, we propose a 3D compression and decompression method. This method also allows the use of the compressed data for a registration process. First, points are selected and grouped, using a 3D-model based on planar surfaces. Next, we use a fast variant of Gaussian Mixture Models and an Expectation-Maximization algorithm to replace the points grouped in the previous step with a set of Gaussian distributions. These learned models can be used as features to find matches between two consecutive poses and apply 3D pose registration using RANSAC. Finally, the 3D map can be obtained by decompressing the models.

1. Introduction

Capturing 3D information has become noticeably more affordable thanks to advances in hardware. The volume of information generated by 3D sensors increases the dimensionality of the problem with respect to 2D devices. Working with 3D information represents an additional advantage in most applications, since more information can be offered to users.

3D data sensors provide an enormous amount of information. It is necessary to develop efficient methods to manage this information under certain time, bandwidth or storage space requirements. A possible strategy could be simply to randomly discard some information obtained or, on the other hand, follow selection criteria. This has the drawback that information potentially useful for later processing of algorithms could be discarded.

One possible strategy could be first to compress the data and then store it or use the compressed information. Working with the compressed information could save processing time and storage. The recovery process could obtain the original information or admit some errors in order to obtain a benefit, for example, getting a better compression rate.

The main contribution in this paper is a method that performs 3D point cloud compression while also enabling the registration of two sets of compressed data. The compression method is designed to be lossy, as decompression does not retrieve the original data. Nevertheless, our previous studies [14] show that losses are within an acceptable margin while compression rates remain high when compared to other methods.

The proposed compression method follows a growing approach in which RGB colored 3D points are successively selected and their surroundings compressed until all the 3D points in the data set are visited. The compressed data is held by Gaussian Mixture Models considering both point position and color information.

Gaussian Mixture Models (GMMs) maintain multi-modal statistical descriptions of data distribution from a local area within the whole 3D scene. Compression is achieved by replacing the raw 3D points from this local area with the corresponding GMM descriptions. Compressed data can be efficiently stored or transmitted. Furthermore, in this paper we demonstrate that GMM descriptions are also valid for matching points between 3D scenes captured from different viewpoints. Thus, using these GMMs it is possible to perform 3D scene reconstruction and even 3DoF pose registration for mobile robotics, which is one of the initial steps for solving 6DoF SLAM.

One of the most researched fields in robotics is the processing

**Corresponding author: Tel.: +34-965-903400 ext: 2992; fax: +34-965-903902;
e-mail: javi.navarrete@ua.es (Javier Navarrete), dviejo@ua.es (Diego Viejo), miguel.cazorla@ua.es (Miguel Cazorla)

of Simultaneous Localization and Mapping (SLAM). Knowing where a robot is and what its environment is like has a high computational cost. Using compressed clouds to this end enables us to make a reconstruction of the full map with a lower cost of storage or transmission. This is a motivation to develop methods specifically designed to allow pose registration as the simplest part of the SLAM processing.

The rest of the paper is organized as follows: Section 2 contains a brief review of common compression methods, and more specifically of 3D data compression methods. In Section 3, we present the proposal for 3D data compression and registration. In Subsection 3.1, we describe how data selection is performed. In Subsection 3.2, we explain how compression works and in Subsection 3.3, we describe how compressed data is used for pose registration. Some experiments and their results are presented in Section 4. Finally, Section 6 presents the conclusions and directions for future work.

2. Related work

3D data compression increases the bandwidth available for data transfer and allows the reduction of physical storage. Both features are desirable in robotics tasks, especially for real time applications.

There are several classifications of compression methods based on loss of information, codification or point clustering. In a classification based on loss of information, there are two types of methods: lossy and lossless. Once a point cloud has been compressed and decompressed, lossless methods return the same original point cloud without any error in point color or coordinates whereas lossy methods usually return point clouds with some errors.

Codification-based compression methods take advantage of the information redundancy commonly found in large streams of data. As an example of this kind of technique, general compression algorithms such as the well-known [23] or [4] can be outlined. These methods are used in compression applications like ZIP and LHA respectively. General data compression is a well-established field of research and we can find highly optimized implementations. Nevertheless, it is difficult to use a general compression method under specific time requirements. Moreover, it is necessary to decompress the data before using it, which is another disadvantage of this kind of method. Furthermore, data sets in which information redundancy is minimal cannot be highly compressed by lossless algorithms.

Some lossy methods are based on space organization. [17, 10, 11, 12] exploit octree space partitioning for achieving the compresion. They need to encode, for each node of the tree, which octree subdivisions are empty and which ones are iteratively subdivided. Thus, in [17] the point-cloud is encoded in terms of occupied octree-cells. [10, 11] propose a method for non-empty child prediction based on local neighborhood, and use Gaussian sphere based normal representation and prediction, producing an entropy reduction and thus better geometry coding performance. [12] uses structure comparison of the octrees obtained from consecutive 3D point sets in a stream in order to remove temporal redundancy and to achieve good compression rates. Instead of using the octree approach, [5] employs a data-driven point clustering approach which iteratively obtains hierarchical levels of detail (LOD). This LOD hierarchy is analysed to obtain a progressive encoding of the original model.

In the literature, we can find a number of techniques based on point clustering using geometric considerations, where a certain loss of information is assumed. [7] uses the eigenvalues from the whole point cloud to extract curvature information, detecting repetitions and removing duplicates. Although this work makes a combination of compression and registration, similar to the proposed method, they use only geometric information. [14] performs a plane extraction, concave hull and Delaunay triangulation to replace co-planar points by planar patches.

Our method is based on Gaussian Mixture Models, and in the literature we can find that GMMs have been used for 2D image compression in [20], and also for other tasks like classification and recognition [22, 19], image segmentation [16], or image matching [13]. Nevertheless, to the best of our knowledge, GMMs have not been used for 3D point cloud compression.

3. Proposed method

In this work, we present a lossy compression method based on point clustering and replacement. The information contained in the compressed data can be used by further processes without its having to be decompressed.

Whatever the source of acquisition or the rendering model used, the three-dimensional information can be represented as a 3D point cloud. A 3D point cloud is a set of 3D points, defined by their spatial coordinates x, y, z, but which may also incorporate other information from the scanned surface such as color, material reflectivity, surface normal vector, etc. We define these 3D point coordinates along with RGB color information. Thus a 3D point is defined as $p = (x, y, z, r, g, b)$, where (x, y, z) are the point coordinates and, (r, g, b) the color components red, green and blue.

As a prior step to compression, points of interest are selected. According to our experiments, these points are the best candidates to perform compression in both Cartesian and RBG color spaces. This step is addressed by planar patch extraction as defined in [21], which efficiently obtains planar descriptions for the planar surfaces in a 3D scene. With this selection, we determine the points to be compressed and those to remain unaltered. We modify the original planar patch extraction method to iterate with different radii in order to increase the number of selected points and, thus, improve the compression ratio.

In the main step, the compression, we use a non-supervised learning algorithm based on Gaussian Mixture Models together with the Expectation-Maximization (EM) algorithm. Specifically we use a fast variant of GMM called FastGMM as described in [9]. The GMMs hold the input data as 3D multivariate Gaussian distributions, which allows the reconstruction of the original data with a certain statistical reliability.

3.1. Data selection

The planar patch calculation is defined in [21]. The aim of this work is to obtain information about planar surfaces from scenes captured by a 3D sensor during a robot movement in order to register them and obtain the robot movement. The amount of data needed to represent each scene as planar surfaces, using patches, is significantly reduced. Hence, the registration time is also reduced. However, this approach is not designed to compress the original data and most of the input data is lost. Nevertheless, the authors use a planarity test that we reuse in our proposal.

Given a point p in a 3D image, this planarity test provides us a quantitative value of how well the neighboring points around p fit to a planar surface. In our proposal, we process all the points in the neighborhood of any point marked as planar by the test. The original method defines a radius for the neighborhood that is related to the distance from the 3D sensor to the scanned point. We follow the same idea but in a k-pass approach. The first iteration obtains all the planar areas in the 3D scene using the originally proposed radius. Subsequent iterations scale down the radius using Equation 1

$$r_k = \begin{cases} r & \text{if } k = 0 \\ 0.414 r_{k-1} & \text{other.} \end{cases} \quad (1)$$

where r is the initial radius used in the selection process, and k is the number of iteration. We begin with $k = 0$ and at each iteration the radius is reduced in order to reach smaller areas at each iteration. We have to select the maximum value of k. For that reason, we have run several experiments to test how compression rate is affected by different initial radius and k values. As shown in Figure 1, for a $k \geq 4$ there is no difference in the compression rate. Thus, we will use $k = 4$ for our experiments.

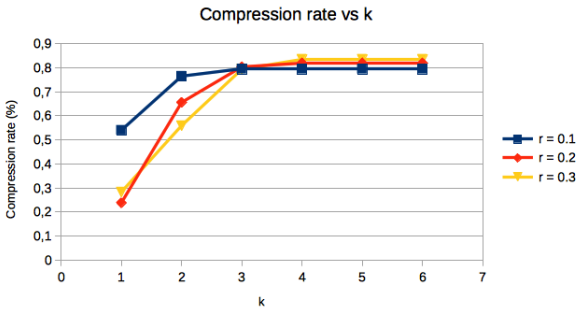

Fig. 1. Compression rates with respect to different number of final k.

The effect of the multi-iteration approach can be observed in Figure 2. We can see in white the areas selected by the planar patches extraction algorithm. The greater the number of iterations the more points are compressed. This allows us to use the number of iterations as a parameter to adjust the degree of compression we want to fulfill.

3.2. Compression

The set of points from each planar patch extracted in the previous step is then compressed using a Gaussian Mixture Model

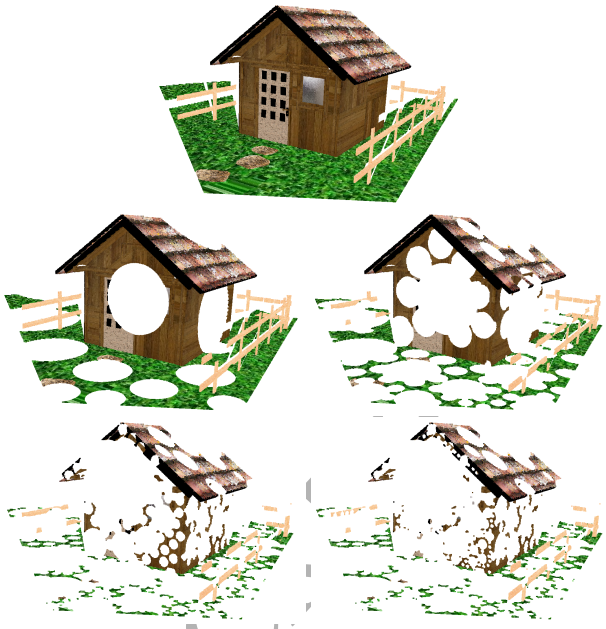

Fig. 2. Point selection using a planar patch approach. Top: Original point cloud. From the second row and from left to right, first to fourth iteration.

(GMM). In general, GMMs are used to adjust K multivariate Gaussian distributions over data in R^n space. As previously stated, in our case the points belong to R^6. Let N be a set of points in R^6, $GMM(N)$ is defined in Equation 2.

$$GMM(N) = \{\{\mu_k\}, \{\Sigma_k\}, P(k)\} \quad (2)$$

where:

- $\{\mu_k\}$, are k 6-dimensional vectors that are the mean of each Gaussian distribution.
- $\{\Sigma_k\}$, are k covariance matrices (6×6), one for each Gaussian distribution.
- $P(k)$, are k probabilities, one for each Gaussian distribution.

The problem here is that the number of k Gaussian Models needed is not known *a priori*. To overcome this, we use the FastGMM method defined in [9]. This variant initializes the adjustment process using $k = 1$. Once the adjustment is finished, the result is evaluated by analysing the age of each Gaussian component, the determinant of the partial covariance matrix, and the log-likehood variations between consecutive iterations. If the adjustment conditions are not fulfilled, a new Gaussian Model is generated from an already existing one and the adjustment process continues. In this method, starting iterations are executed very quickly, due to the low number of distributions, while the latest iterations present a similar time consumption to generic GMMs. Thus, limiting the maximum value of k, it is possible to achieve a significant time acceleration with a low impact on GMM adjustment.

Finally, the compression of a subset of 3D points obtained from the input image by the planar patch extraction process described above and defined by a GMM is achieved by replacing

these points with the planar patch description and the Gaussian components in the GMM:

$$S \equiv \{px_i, py_i, pz_i, nx_i, ny_i, nz_i, |S|, r_i, th_i, \{\mu_k\}, \{\Sigma_k\}, P(k)\} \quad (3)$$

where:

- px_i, py_i, pz_i, are the coordinates of the geometric center of the planar patch.
- nx_i, ny_i, nz_i, is the normal vector of the surface composed by S.
- $|S|$, number of points of S .
- r_i, is the radius used to get the planar patch.
- th_i, is the *thickness* of the planar patch.
- $\{\mu_k\}$, is a set of k 6-dimensional vectors. Each one is the mean of a Gaussian distribution in the GMM.
- $\{\Sigma_k\}$, is a set of k covariance matrices (6×6), one for each Gaussian distribution in the GMM.
- $P(k)$, is a vector of k elements, where each element is the probability of the corresponding Gaussian distribution in the GMM.

Furthermore, we consider whether compressed data result in a smaller size than the original point set in the patch. When the benefit is not clear, the patch is not compressed and its points are included in the resulting 3D image without modification. At this point, we also take into account some special situations for the adjusted GMM. Complete maladjustment or insufficient number of points in the patch also resulted in the patch being discarded. When this happens, points are inserted in the uncompressed list. This does not improve the compression ratio but does not increase decompression error, either.

Until now, the process described allows us to compress a colored 3D point set. Another feature of our proposal is that the process can, to a certain degree, be undone. The compressed data includes the information for each planar patch and a set of multivariate distributions resulting from the GMM adjustment. To decompress, we need to sample these distributions. From Equation 3 we have that a compressed planar patch includes the number of points included in patch $|S|$ and its proportion to be generated from each multivariate Gaussian distribution μ_k, Σ_k. In order to sample a vector x giving a multivariate Gaussian distribution with N dimensions, mean μ_k and covariance matrix Σ_k we perform [8]:

1. Find a real matrix A that accomplishes $AA^T = \Sigma_k$.
2. Generate a vector $z = (z_1, ..., z_n)^T$ where its coordinates are obtained from N independent normal standard distributions. These coordinates can be generated using the Box-Muller transform[3].
3. Finally, $x = \mu_k + Az$. Each vector x is an uncompressed 3D point.

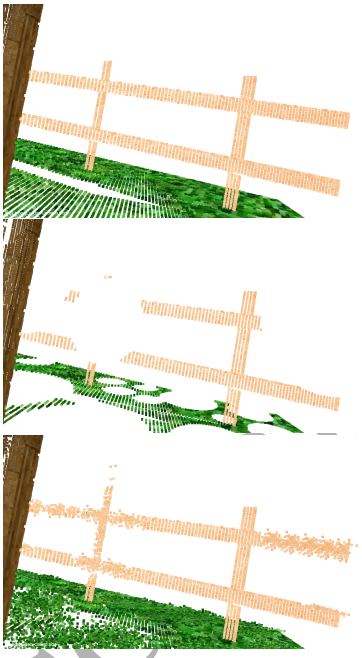

Fig. 3. Example of a synthetic point cloud (top), without the first patches (middle) and after reconstruction (bottom).

Fig. 4. Example with a real point cloud (top), without the patches (middle) and after reconstruction (bottom).

Repeating steps 2 and 3 we obtain samples from a multivariate Gaussian distribution. We apply this procedure to all Gaussian components within a planar patch. Finally, we do the same with all the compressed patches in order to get the complete uncompressed point cloud.

Figure 3 shows the effects of the compression-decompression

process on a point cloud captured in a synthetic way. The white spheres represent the information extracted from the point cloud and replaced by the Gaussian mixture. Reconstruction shows the adaptation of the distributions to fit geometrical shapes as in the case of the fence. In Figure 4, this same process is applied to a Kinect capture and some loss in reconstruction can be observed, although the quality is valid enough for many applications.

3.3. Pose Registration

We also demonstrate that the compressed information can be useful for tasks such as pose registration since the GMMs include additional information about the scene. Thus, the compressed patches from different scenes are utilized as the features used for matching the common parts of two 3D scenes. Then, Random Sample Consensus (RANSAC,[6]) is used to obtain the transformation that aligns the two scenes.

First, we perform a search to find all the correspondences between features from both point clouds $N1$ and $N2$. To establish the correspondences we define three different distances. Given features consisting in two compressed planar patches $S_{N1} \in N1$, and $S_{N2} \in N2$:

1. Angle distance between normal of S_{N1} and normal of S_{N2}.

$$d_\alpha(S_{N1}, S_{N2}) = acos\left(\frac{S_{N1} \cdot S_{N2}}{\|S_{N1}\|\|S_{N2}\|}\right) \tag{4}$$

   where $S_{N1} \cdot S_{N2}$ is the dot product between the normals and $\|\cdot\|$ is the L2 norm.
2. Euclidean distance between both geometric centers of S_{N1} and S_{N2}.

$$d_e(S_{N1}, S_{N2}) = \sqrt{(p_{x_1} - p_{x_2})^2 + \left(p_{y_1} - p_{y_2}\right)^2 + (p_{z_1} - p_{z_2})^2} \tag{5}$$

   where $p_{x_i}, p_{y_i}, p_{z_i}$, are center coordinates of the patch S_{Ni}.
3. Distance between Gaussian components of both planar patches. This value is based on the Bhattacharyya [1] distance and a selection process to get the final distance $d_P(S_{N1}, S_{N2})$.
   The *Bhattacharyya* distance measures the similarity of two probability distributions. For two multivariate distributions C_1, C_2, it is defined by:

$$d_{bha}(C_1, C_2) = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \frac{1}{2} ln\left(\frac{det\Sigma}{\sqrt{det\Sigma_1 det\Sigma_2}}\right) \tag{6}$$

   where:

   - μ_1, μ_2, are the mean vectors of C_1, C_2. In our case, 6-dimensional vectors.
   - Σ_1, Σ_2, are the covariance matrix 6×6.
   - $\Sigma = \frac{\Sigma_1 + \Sigma_2}{2}$.

   We have to consider that two patches can have several Gaussian components and the number could be different. Then, we perform the next evaluation:

   3..1 Given a Gaussian component C_i of a planar patch S_{N1}, we get all the *Bhattacharyya* distances between all components.
   $d_{bha}\left(C_i, C_j\right) \; \forall \; C_i \in S_{N1}, C_j \in S_{N2}$
   3..2 We locate for C_i the two closest C_j.
   $d_{bha_{min}} = \min\{d_{bha}\left(C_i, C_j\right)\}$ and
   $d2_{bha_{min}} = \min\{\{d_{bha}\left(C_i, C_j\right)\} - d_{bha_{min}}\}$
   3..3 We discard matches that do not accomplish (we can manage a factor of 0.2 in order to make matches more or less permissive): $d_{bha_{min}} < d2_{bha_{min}} * 0.2$
   3..4 We check the distance in both directions.
   $d_{bha_{min}}\left(C_i, C_j\right) = d_{bha_{min}}\left(C_j, C_i\right)$
   3..5 We obtain the mean distance for non-discarded distances. This mean is our third distance $d_P(S_{N1}, S_{N2})$. Moreover we add an additional condition in our implementation, to ensure that all the compressed planar patches have at least three Gaussian distributions in order to obtain a better matching.

The final distance between two patches is computed as a weighted combination of these:

$$DF = \beta_1 d_\alpha + \beta_2 d_e \; + \beta_3 d_P \tag{7}$$

In general, the coefficients $\beta_1, \beta_2, \beta_3$ must ensure that three distances have a similar weight in the final distance. Thus, their values can be adjusted to accommodate different kinds of 3D sensors. For example, sensors with a high sensing range, such as 3D lasers, need low values for $\beta 2$. In the case of experiment in 6 we used $\beta_1 = 1.0, \beta_2 = 0.0001, \beta_3 = 0.5$. However, for sensors with a low capturing range, such as Kinect device, a value of $\beta_2 = 1.0$ is preferred.

DF is computed for each patch from one scene to each patch from the other. In order to improve the matching, two patches S and M are matched if $DF(S, M) < 0.2 * DF(S, N) \forall N \neq M \in$ the second scene. Finally, when all the correspondences are established we run the RANSAC algorithm in order to obtain the transformation that aligns the two scenes.

4. Experiments and results

We have used the dataset and tools in [15] to test the proposed method. This benchmark provides 104 point clouds and several tools for comparing compression methods. It presents results for different structure and texture levels. Using this benchmark, we compare our method with one lossless and two other lossy methods. LZ77 [23] is chosen as an example of a lossless method. The other lossy methods are: Octree [12] (using 24 bits) and Morell2014 [14] (with k=1 and k=5). The former is fast and able to achieve high compression rates and the latter uses geometric information of the environment (mainly planes) to get even higher compression rates.

In Figure 5, we observe how our method achieves higher compression ratios for real data than all the other methods. Analysing distance and color errors, as it is a lossless method, the LZ77 has neither distance nor texture errors. However, this method is not able to achieve higher compression rates. The lossy method with fewest errors, the Octree method, has an

almost insignificant number of errors. Our method has fewer errors than the other lossy method, but more than the Octree one. However, the proposed method is able to get higher compression rates than the Octree and LZ77, and, of course, the Morell2014 method. We have to keep in mind that besides compressing data our method is able to perform registration with the compressed data, which is an interesting property none of the other methods have.

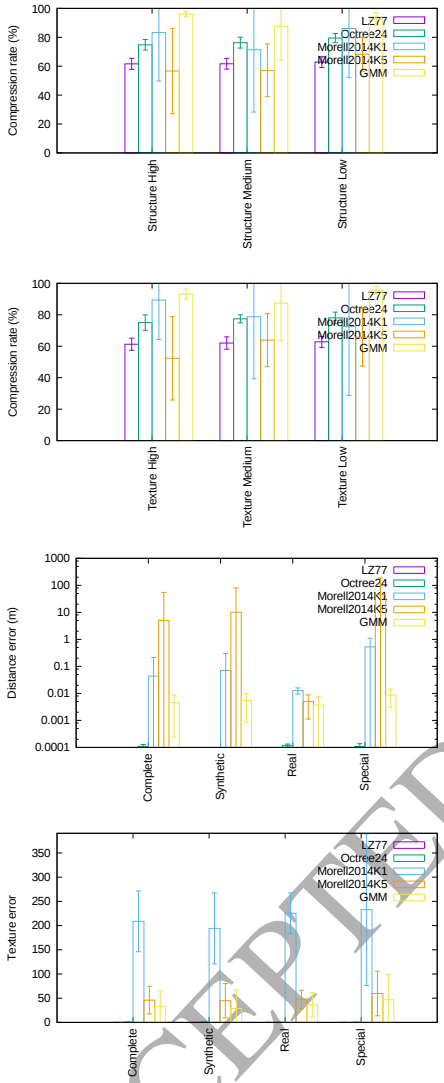

Fig. 5. Compression rate using real data w.r.t. structure (first) and texture (second). Comparison of distance error (third) and color error (fourth) using complete data set.

Once the compression procedure has been tested, we check the pose registration procedure. We perform two experiments using different capture sensors. The only previous operation performed with the point clouds is compression. We compare our method with that proposed in [18]. This method also uses the planar patch extraction method.

The first experiment uses data captured with a Riegl VZ-400 and a thermal infrared camera [2]. This is a large and precise dataset called "thermobremen" with 11 poses separated by distances between 20 until 40 m, with a total of 40.7 million of points. This dataset is also used in [21], and thus we can compare our results with this approach as shown in Table 1. After compression, we obtain a mean of 1048 patches/pointcloud, each with 5996 points. We achieve a compression rate of 85.25%. Visual results are shown in Figure 6, with a complete reconstruction of some streets of the Bremen city centre. The white circles on the asphalt are the robot positions when data is captured. Some details show us the registry errors that occur such as the roofs of some buildings and especially the lower right wall, where the first and last capture converge. Nevertheless, it is shown that the method can register point clouds assuming a certain error.

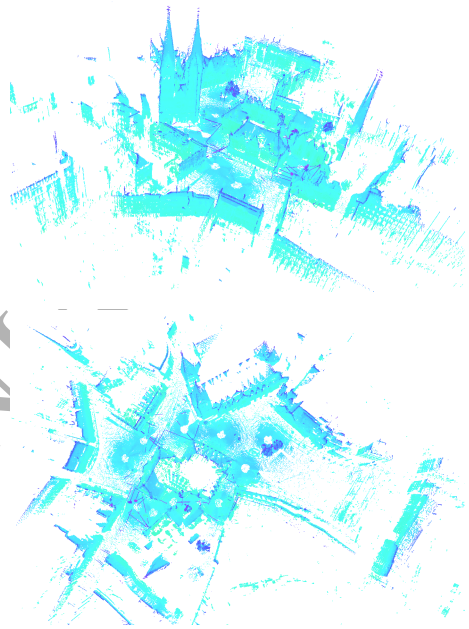

Fig. 6. Example of registration using compressed data.

We spent more time on the patch extraction due to the GMM calculation process. However, our method is able to achieve a lower registration time and fewer translation and rotation errors, as we show in Table 1. The high temporal cost, could be drastically reduced due to the independence of the data in each planar patch and the high degree of parallelism that can be achieved using GPU's.

Table 1. Results of our method compared to those of the method proposed by Viejo et al. [21] using the Thermobremen dataset. Ex.T. is the extraction time, Reg.T. is the registration time, Tr.E. the translational error and Rot.E. the rotational error.

	N. Pat.	Ex.T.	Reg.T.	Tr.E.	Rot.E.
Viejo et al.	2,208	1.6s	54.8s	0.89m	1.27^o
Proposal	1,048	240.3s	3.1s	0.32m	1.01^o

For the second experiment we have used two sequences captured with the Kinect sensor called *fr2_desk* and *fr1_360*. This is part of the [18] dataset for SLAM benchmarking. This dataset has currently 90 sequences, two of them have been selected for their movements in order to check the effects they have on results. Both have been captured with the device in hands, giving great variety of movements. Freiburg2-desk contains all kind

Table 2. Benchmark output for pose relative error. Freiburg2-desk data set.

	proposed	Only ICP	Vis. Feat.
compared pose pairs	2080 pairs	2231 pairs	2231 pairs
transl. error.rmse	0.014 m	0.032 m	0.023 m
transl. error.mean	0.013 m	0.009 m	0.007 m
transl. error.median	0.012 m	0.006 m	0.006 m
transl. error.std	0.007 m	0.031 m	0.028 m
transl. error.min	0.001 m	0.000 m	0.00 m
transl. error.max	0.077 m	1.30 m	1.30 m
rot. error.rmse	0.64 deg	1.03 deg	0.96 deg
rot. error.mean	0.57 deg	0.30 deg	0.31 deg
rot. error.median	0.01 deg	0.01 deg	0.00 deg
rot. error.std	0.31 deg	0.95 deg	0.91 deg
rot. error.min	0.04 deg	0.013 deg	0.02 deg
rot. error.max	2.41 deg	42.42 deg	42.42 deg

Table 3. Benchmark output for pose relative error. Freiburg1-360 data set.

	proposed	Only ICP	Vis. Feat.
compared pose pairs	743 pairs	736 pairs	736 pairs
transl. error.rmse	0.035 m	0.024 m	0.021 m
transl. error.mean	0.021 m	0.017 m	0.016 m
transl. error.median	0.010 m	0.012 m	0.013 m
transl. error.std	0.027 m	0.017 m	0.013 m
transl. error.min	0.000 m	0.000 m	0.001 m
transl. error.max	0.106 m	0.156 m	0.094 m
rot. error.rmse	2.33 deg	0.79 deg	0.645 deg
rot. error.mean	2.08 deg	0.61 deg	0.557 deg
rot. error.median	0.03 deg	0.01 deg	0.01 deg
rot. error.std	1.04 deg	0.49 deg	0.32 deg
rot. error.min	0.05 deg	0.03 deg	0.06 deg
rot. error.max	8.80 deg	3.95 deg	2.65 deg

of motions while Freiburg1-360 generates a panoramic view where the main motion of the sensor is rotation. The first sequence has 69.15s of duration and 18.88m trajectory length. Table 2 shows the comparison with two of the registration methods provided in the data set. In this case, we obtain less relative error that achieved with the other results. The data for the second sequence is 28.7s and 5.818m. The results are shown in Table 3. Now, our method provides more error than the other two methods. Once again, although our method has in some cases more registration errors than other methods, the combination of compression and registration make our method an interesting tool for processing 3D point clouds.

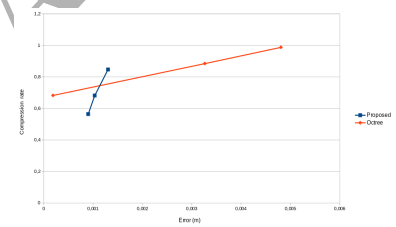

Fig. 7. Compression rate and error study for Octree algorithm and our method.

Fig. 8. Some examples for study of decompressed points

Figure 7 shows a position study of decompressed points, showing the compression rate and position error of the Octree algorithm and our proposal. It is possible to visualize how the compression rate and mean error evolve using different compression profiles in each method. For Octree, the default high/medium/low profiles included in the Point Cloud Library have been used, and without using any process to reduce the number of points. Parameters have been selected incrementally for our method to demonstrate that it can also achieve similar compression ratios with similar or lower errors. We are aware that the parameterization of both methods can be adjusted for each specific point cloud to achieve better compression rate or quality results. In the test multiple captures have been taken as input data, all of them using Kinect. Some examples are shown in Figure 8.

5. Discussion

In this section, we proceed to analyse the advantages and disadvantages of the proposed method, compared with other state-of-the-art methods.

The main advantage of the proposed method is that we are able to compress a 3D point cloud and, using that compressed cloud, make 3D registration. Working with compressed point clouds is useful because this kind of information is usually huge. Although other work [7] makes a combination of compression and registration of 3D data, the proposed work not only uses geometric information but also color data.

Regarding the proposed compressing method, it obtains better compression rates than state-of-the-art methods, under several configurations. Another of the advantages of using mixtures of Gaussians for replacing a point cloud is that one can generate arbitrarily dense clouds regardless of the resolution of the sensor that originally captured the cloud.

Regarding the registration process, it provides similar or fewer errors than other standard methods. But using the combination of both, compression and registration, makes the proposed method a good approach when the volume of information is huge. Other registration methods, like ICP, are not able to manage huge volume of data.

The proposed method has also a set of disadvantages. The main one is that the method is not able to recover the initial point cloud. This is due to the lossy process inherent in the method. This could be a problem if original point cloud must be recovered, but it could not be dramatical for other kind of applications, like robot navigation.

It also loses meta-information, like the neighbourhood information in a RGBD camera or a 3D sensor which processes information by linear scans. This can be used to infer topological relationships in the point cloud, but after replacing the raw points with mixtures of Gaussians this is no longer possible.

Besides, the Gaussians will be especially bad approximations over discontinuous parts of the scene (e.g. corners, edges), but those are often the parts of the scene that are most interesting. In the reconstruction these parts will be reproduced poorly as is clear from the included images. However, the use of a planar patch mitigates that problem. As it can be seen in Figure 4, the method is able to recover the fence with a good result.

6. Conclusions and future work

In this paper, we propose a compression method for colored 3D images that uses planar surface analysis in order to identify the areas of the 3D image to be compressed. Then, we propose the use of Gaussian Mixture Models (GMMs) for reducing the size of the data. We use a fast algorithm for adjusting GMMs to the points contained in each planar patch previously extracted from the input 3D image. Our compression proposal has been tested using a public compression dataset and has been compared to several methods for 3D data compression. We obtain significantly better compression rates, especially for real data, and with reasonable errors for most applications.

We have also developed a method that performs 3D point set registration, using the compressed information obtained by our compression approach. We have designed a new distance measurement in order to set the correspondences between compressed patches from different scenes. This distance is used to find matches between patches and from those matches the RANSAC algorithm computes the transformation that aligns the input scenes. The validity of the method has been demonstrated in two experiments that show promising results.

As a future work for compression, we consider using alternative methods of data selection to evaluate how the different methods can affect the compression result. We also consider developing a parallel implementation of the compression method using Graphics Processors Unit, GPUs) in order to speed up the compression process. We plan to study whether improvements would occur in time and quality of results by adding a test of goodness when we fit the data to Gaussian distributions, by admitting only data sets that exceed a certain value of goodness. Moreover, we are planning to develop the concept of planar patch and Gaussian mixture as a feature descriptor. This has been shown to be valid for pose registration. Hence, it is more than plausible to use it for other tasks such as object recognition.

Acknowledgments

This work has been supported by the Spanish Government TIN2016-76515-R Grant, supported with Feder funds.

References

[1] Bhattachayya, A., 1943. On a measure of divergence between two statistical population defined by their population distributions. Bulletin Calcutta Mathematical Society 35, 99–109.

[2] Borrmann, D., Elseberg, J., Nüchter, A., 2013. Thermal 3d mapping of building facades, in: Intelligent Autonomous Systems 12. Springer, pp. 173–182.

[3] Box, G.E., Muller, M.E., et al., 1958. A note on the generation of random normal deviates. The annals of mathematical statistics 29, 610–611.

[4] Burrows, M., Wheeler, D.J., 1994. A block-sorting lossless data compression algorithm. Technical Report. Digital Equipment Corporation.

[5] Fan, Y., Huang, Y., Peng, J., 2013. Point cloud compression based on hierarchical point clustering, in: Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific, pp. 1–7.

[6] Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381–395.

[7] Friedman, S., Stamos, I., 2013. Online detection of repeated structures in point clouds of urban scenes for compression and registration. International Journal of Computer Vision 102, 112–128.

[8] Gentle, J.E., 2009. Computational statistics. volume 308. Springer.

[9] Greggio, N., Bernardino, A., Laschi, C., Dario, P., Santos-Victor, J., 2012. Fast estimation of gaussian mixture models for image segmentation. Machine Vision and Applications 23, 773–789.

[10] Huang, Y., Peng, J., Kuo, C.C.J., Gopi, M., 2006. Octree-based progressive geometry coding of point clouds, in: Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. pp. 103–110. URL: http://dx.doi.org/10.2312/SPBG/SPBG06/103-110, doi:10.2312/SPBG/SPBG06/103-110.

[11] Huang, Y., Peng, J., Kuo, C.C.J., Gopi, M., 2008. A generic scheme for progressive point cloud coding. IEEE Transactions on Visualization and Computer Graphics 14, 440–453. doi:10.1109/TVCG.2007.70441.

[12] Kammerl, J., Blodow, N., Rusu, R., Gedikli, S., Beetz, M., Steinbach, E., 2012. Real-time compression of point cloud streams, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 778–785.

[13] Li, P., Wang, Q., Zhang, L., 2013. A novel earth mover's distance methodology for image matching with gaussian mixture models, in: Computer Vision (ICCV), 2013 IEEE International Conference on, IEEE. pp. 1689–1696.

[14] Morell, V., Orts, S., Cazorla, M., Garcia-Rodriguez, J., 2014. Geometric 3d point cloud compression. Pattern Recognition Letters 50, 55 – 62.

[15] Navarrete, J., Morell, V., Cazorla, M., Viejo, D., García-Rodríguez, J., Orts-Escolano, S., 2016. 3dcomet: 3d compression methods test dataset. Robotics and Autonomous Systems 75, 550–557.

[16] Nguyen, T.M., Wu, Q., 2013. Fast and robust spatially constrained gaussian mixture model for image segmentation. Circuits and Systems for Video Technology, IEEE Transactions on 23, 621–635.

[17] Schnabel, R., Klein, R., 2006. Octree-based point-cloud compression, in: Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. pp. 111–121. URL: http://dx.doi.org/10.2312/SPBG/SPBG06/111-120, doi:10.2312/SPBG/SPBG06/111-120.

[18] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D., 2012. A benchmark for the evaluation of rgb-d slam systems, in: Proc. of the International Conference on Intelligent Robot Systems (IROS).

[19] Sung, J., Ponce, C., Selman, B., Saxena, A., 2012. Unstructured human activity detection from rgbd images, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE. pp. 842–849.

[20] Van Den Oord, A., Schrauwen, B., 2014. The student-t mixture as a natural image patch prior with application to image compression. The Journal of Machine Learning Research 15, 2061–2086.

[21] Viejo, D., Cazorla, M., 2014. A robust and fast method for 6dof motion estimation from generalized 3d data. Autonomous Robots 36, 295–308.

[22] Wang, M., Gao, Y., Lu, K., Rui, Y., 2013. View-based discriminative probabilistic modeling for 3d object retrieval and recognition. Image Processing, IEEE Transactions on 22, 1395–1407.

[23] Ziv, J., Lempel, A., 1977. A universal algorithm for sequential data compression. Information Theory, IEEE Transactions on 23, 337–343.